5T

УДК 004.272.43:004.274:004.382.2

В.А. Торгашев, И.В. Царёв
Санкт-Петербургский институт информатики и автоматизации Российской академии наук, г. Санкт-Петербург, Россия
civ@mail.iias.spb.su.

Семейство суперкомпьютеров с динамической архитектурой – концептуальные основы

В статье рассматривается архитектура и основные характеристики нового типа суперкомпьютеров, основанных на теории Динамических Автоматных Сетей (ДАС) и идеологии ранее разработанных Мультипроцессоров с Динамической Архитектурой (МДА), позволяющих конструировать высокоэффективные и надёжные вычислительные системы в широком диапазоне производительности. Приводятся характеристики некоторых технических и программных решений.

Введение

Целью данной работы является разработка концептуальных основ и некоторых технических решений для реализации семейства Суперкомпьютеров с Динамической Архитектурой (СДА) в широком диапазоне производительности – от 2 – 6 терафлопс до десятков и сотен петафлопс, с автоматизацией распараллеливания программ, при этом архитектура СДА ориентирована не на конкретные аппаратные решения, но на структуру выполняемой задачи.

На основе теории динамических автоматных сетей (ДАС), разработанной в лаборатории распределённых вычислительных структур Санкт-Петербургского института автоматики и автоматизации РАН в 80-х годах [1], [2], в конце 80-х и в середине 90-х годов прошлого века был разработан ряд образцов мультипроцессоров с динамической архитектурой (МДА).

Существенный прогресс в области элементной базы, в частности появление и постепенное совершенствование ПЛИС, позволяет на данном этапе перейти к разработке и реализации суперкомпьютеров с динамической архитектурой (СДА), сохраняющих основные достоинства МДА, но в то же время позволяющих достичь уровня, сопоставимого с наиболее мощными современными суперкомпьютерами из списка «Top-500», или даже превышающего их по производительности.

Динамические автоматные сети как концептуальная основа МДА и СДА

Любая программа (или любой вычислительный процесс) в МДА (или СДА) представляется в виде динамической автоматной сети (ДАС), элементами (узлами) которой являются автоматы, реализующие функции семи основных классов программных структур: *операторы, данные, отношения, ссылки, ресурсы, типы и структуры (подсети)*, в каждом классе может существовать некоторое множество конкретных типов. Каждый автомат может изменять свои связи с окружающими его автоматами, порождать новые автоматы, а также самоуничтожаться после завершения выполнения своих функций. Класс и тип автомата определяет не только его функции, но и особенности поведения в сети и воздействия на сеть.

На уровне программирования автоматы представляются объектами, реализуемыми на специальном графическом объектно-ориентированном языке программирования ЯРД [3]. Программа на этом языке является не алгоритмом, т.е. заранее определённой последовательностью действий, а всего лишь описанием начальной структуры сети (и входящих в неё подсетей), а также правил её преобразования, впрочем, основная часть этих правил изначально заложена в свойствах объектов (автоматов) разных классов и в большинстве случаев не требует дополнительных описаний (если только некоторый объект не обладает какими-то особенными свойствами и поведением в сети, которые требуют дополнительных описаний).

В процессе вычислений ДАС постоянно изменяет свою конфигурацию, генерируются (порождаются) новые автоматы, уничтожаются автоматы, выполнившие свою функцию, могут изменяться связи между автоматами, а также автоматы могут изменять свой собственный тип и класс, не говоря уже о значениях данных, их структуре и объёме. Это называется автотрансформацией ДАС. Но в любой момент вычислительного процесса текущее состояние ДАС соответствует структуре задачи на данном этапе её решения (т.е. некоторому множеству объектов и связей между ними), таким образом архитектура МДА/СДА динамически подстраивается под структуру решаемой задачи, причём каждому объекту (автомату) в процессе выполнения сопоставляется некоторый виртуальный процессор.

В идеале для каждого объекта программы (автомата) должен динамически создаваться процессор с единственной функцией, соответствующей свойствам данного объекта (например, функция процессора, соответствующего автомату класса «данные», – всего лишь хранение некоторых значений и описаний их типов, структур, свойств и состояния, т.е. это – некоторый объём памяти и некоторая структура данных, а функция объекта класса «оператор», типа «скалярное умножение векторов», – это совокупность умножителя, сумматора и некоторой достаточно простой управляющей схемы). После выполнения своей функции (например, данные уже не нужны для продолжения решения или оператор скалярного умножения выполнил свою функцию) соответствующий процессор уничтожается, а на его месте (в смысле физической аппаратуры) может быть создан новый процессор для выполнения других функций, присутствующих в программе.

На практике в настоящее время вполне возможна реализация таких виртуальных процессоров посредством динамического перепрограммирования ПЛИС, на основе которых они и реализуются, однако пока такой процесс не обеспечивает достаточной эффективности (время перепрограммирования ПЛИС сопоставимо со временем выполнения функции конкретного виртуального процессора), поэтому на данном этапе перепрограммирование процессоров возможно эффективно реализовать только на стадии загрузки очередной задачи в систему. В этом случае ПЛИС перепрограммируются на некоторый набор операторных (процессорных) функций, соответствующий конкретной задаче, на требуемую для данной задачи разрядность и форму представления чисел или других данных, например, на логические, текстовые операции и данные или арифметические операции с фиксированной или плавающей точкой, и на числа с разрядностью 8, 16, 32, 64 бит или более, если это необходимо для решения конкретной задачи.

Управление процессом распараллеливания и параллельного выполнения задачи определяется типом и состоянием объектов программы (автоматов), а распределение вычислений между аппаратными ресурсами (вычислительными модулями) – свойствами коммутационной подсистемы, которая обеспечивает «интеллектуальную маршру-

тизацию». Любой объект программы (автомат) и соответствующий ему виртуальный процессор может быть направлен для выполнения своих функций в любой вычислительный модуль системы, при этом будут учитываться как параметры загрузки произвольного модуля (процессора), так и наличие, пропускная способность и текущая загрузка любых возможных путей связи между вычислительными модулями. В любом случае выбирается оптимальный путь передачи данных и программ (виртуальных процессоров) между модулями.

Отличительные особенности вычислительных систем с динамической архитектурой

Основными особенностями МДА/СДА (далее везде используется аббревиатура СДА, подразумевающая в то же время соблюдение основных принципов МДА), отличающими их от других мультипроцессорных систем, являются:

– существенное (в десятки раз) уменьшение объёма аппаратуры, энергопотребления и стоимости при сопоставимых характеристиках производительности;

– представление программ не в виде алгоритма, а в виде сети из объектов;

– автоматическое динамическое распараллеливание задач, не требующее вмешательства программиста, при этом программист освобождается и от необходимости учитывать в программе количество и характеристики имеющихся в системе вычислительных ресурсов;

– полностью децентрализованное и асинхронное управление параллельными вычислительными процессами;

– аппаратное разделение функций вычисления (исполнения), управления и коммутации между соответствующими процессорами (ИП – исполнительный процессор, УП – управляющий процессор и КП – коммутационный процессор), которые имеются в каждом вычислительном модуле (ВМ), что существенно увеличивает эффективность вычислений, поскольку ИП освобождается от выполнения управляющих и коммутационных функций;

– крайне малый объём (десятки килобайт) операционной системы, часть функций которой реализуется аппаратно, а часть представлена методами объектов программы;

– наличие высокоскоростных многоразрядных каналов связи между вычислительными модулями, обеспечивающих благодаря интеллектуальным функциям КП автоматическую многовариантную маршрутизацию потоков данных и программ [4];

– отсутствие ограничений на количество аппаратных ресурсов, включая и вычислительные модули, что позволяет неограниченно масштабировать систему без внесения каких-либо изменений как в системное программное обеспечение, так и в прикладные программы;

– высокая надёжность вычислений и информационная безопасность – решение задачи завершается даже при множественных отказах ВМ, следствием которых является только некоторое уменьшение производительности, а специальная система адресации программных объектов не допускает несанкционированного доступа к объектам программы, что практически исключает возможность существования вирусов и других вредоносных программ [5], [6].

Остановимся на некоторых из вышеперечисленных особенностей, играющих ведущую роль в достижении столь высоких характеристик СДА. Те вопросы, которые ниже не рассматриваются, рассмотрены в публикациях, на которые имеются ссылки.

В первую очередь, возникает естественный вопрос: «Каким образом достигается столь существенное уменьшение габаритов, энергопотребления и стоимости?». Прежде всего это достигается за счёт более полного использования аппаратуры в каждый момент времени выполнения вычислительного процесса.

Во-первых, в системе не используются типовые процессоры, архитектура которых весьма сложна, но значительная часть аппаратуры которых либо выполняет вспомогательные функции (дешифрация сложных по структуре команд, организация конвейеров, кэш-памяти, предсказание переходов и многое другое), либо не используется при выполнении некоторой конкретной операции, поскольку сложная универсальная система команд таких процессоров предполагает и наличие арифметического устройства (АУ) большой сложности, значительная часть которого в каждый конкретный момент времени простаивает.

В СДА для каждой конкретной задачи (или даже фрагмента задачи) создаётся виртуальный исполнительный процессор, содержащий в АУ только те фрагменты схем, которые способны выполнять ограниченный набор операций, необходимых в данной задаче, и только для используемых в данной задаче типов и разрядностей данных. Таким образом, в каждый момент времени выполнения вычислительного процесса, оказывается загруженной бòльшая часть аппаратуры, в то же время вспомогательные функции крайне просты и не используют сколько-нибудь существенной части аппаратуры.

Во-вторых, разделение функций между специализированными ИП, УП и КП освобождает более сложный ИП от выполнения функций организации вычислительного процесса, обеспечивая максимальную загрузку аппаратуры ИП, в то время как функции УП и КП достаточно просты и они не занимают большого объёма аппаратуры, при этом они работают параллельно с ИП.

В-третьих, использование относительно дешёвых ПЛИС, к тому же работающих на сравнительно невысоких тактовых частотах (в пределах 200 – 400 МГц), существенно понижает и стоимость системы в целом, и энергопотребление, и, что важно, существенно снижает требования к системе охлаждения.

Тогда, естественно, возникает вопрос, каким образом многопроцессорная система, работающая на столь низких по современным понятиям частотах, способна достичь столь высоких результатов в производительности? Здесь следует обратиться к самой природе, которая создала столь компактный (всего около 1,5 литров объёма), универсальный и совершенный «суперкомпьютер», как человеческий мозг, который, работая на частотах на несколько порядков меньших, чем используемые в современной вычислительной технике, тем не менее способен практически мгновенно выполнять столь сложные и интеллектуальные задачи, как распознавание образов, анализ сцен, принятие решений, ориентация в пространстве, управление движением сложного по структуре тела и многие другие задачи, которые явно не по силам самым современным суперкомпьютерам (и роботам). Конечно, СДА не претендует пока на соревнование с возможностями мозга, однако основной структурный принцип – наличие большого количества параллельно работающих простых по структуре процессоров (как в мозге – нейронов) – здесь в какой-то мере соблюдается.

Следующий вопрос, который может возникнуть у читателя, – каким образом достигается столь малый объём операционной системы (ОС)? На самом деле функции любой современной ОС разделяются на несколько частей: управление вычислительными процессами (в данном случае – параллельными), управление внешними устройствами (включая и накопители на разного рода дисках), человеко-машинный интерфейс (в том числе графическое или текстовое отображение информации на экране монитора, реакция на интерфейсные устройства – клавиатуру и мышь) и т.д.

В СДА реализуется только первая из перечисленных функций, а остальные возлагаются на «хост-машину», т.е. на обычную ПЭВМ, выполняющую по сути функцию «интеллектуального терминала», которая может работать под управлением любой из существующих ОС – Windows, Linux и т.п. Внешние устройства не подсоединяются непосредственно к вычислительным модулям СДА, соответственно не нужно управление этими устройствами (т.е. не нужны какие-то специфичные драйверы), не нужна и обработка прерываний (в самих модулях СДА прерывания не используются – как аппаратные, так и программные). В то же время сами функции управления распараллеливанием вычислительного процесса и выполнением отдельных ветвей этого процесса, выполняемые в соответствии с моделью ДАС, крайне просты и в значительной мере могут быть выполнены аппаратным способом в виде соответствующих функций УП и КП. Кроме того, на хост-машине могут быть выполнены части программы, некритичные к производительности и не требующие параллельного выполнения.

И, наконец, ещё один вопрос, который следует рассмотреть, – каким образом может быть организовано неограниченное количество аппаратных ресурсов (модулей) и соответственно неограниченное масштабирование системы. Каждый ресурс СДА (вычислительный модуль, блок памяти и т.д.) имеет однозначную адресацию (нумерацию), выраженную одним 32-разрядным машинным словом, либо последовательностью таких слов. Для собственно адресации используется 31 младший разряд, а старший разряд 32-разрядного слова содержит признак того, является ли оно (слово) последним в «адресе» (0) либо за ним следует продолжение (1). Но и одного такого слова достаточно для нумерации 2^{31} аппаратных ресурсов, что составляет более 2 миллиардов, явно в обозримом будущем какая-либо вычислительная система вряд ли будет обладать таким количеством ресурсов (модулей, процессоров), даже с учётом малого объёма процессоров СДА.

Некоторые вопросы реализации СДА

Основой реализации любого варианта СДА является так называемый «технологический модуль» (ТМ), в рассматриваемом варианте состоящий из шести плат, конструктивно объединённых в виде «этажерочной конструкции».

Три платы – одинаковы и содержат по 32 исполнительных модуля (ИМ), выполненных на основе ПЛИС «Altera Cyclon». Каждый ИМ включает в себя ИП и КП, реализованные в рамках одной ПЛИС.

Одна плата является управляющим модулем (УМ) и включает в себя управляющий процессор (УП) и 4 ИМ. Каждый ИМ, а также УП и ИМ снабжены памятью объёмом не менее 128 Мб на процессор (все блоки памяти независимы и предназначены для использования только одним процессором). Кроме того, имеются две небольшие вспомогательные платы (источники питания и пр.).

Каждый ИМ и УП имеют по 4 высокоскоростных канала для связи с соседними модулями. Пропускная способность канала – 3,6 Гбайта в секунду при разрядности 32 бита.

Кроме собственно вычислительных модулей с динамической архитектурой, каждый ТМ имеет в своём составе SOM (System On Module) и жёсткий диск (винчестер) с ёмкостью не менее 500 гигабайт, которые конструктивно присоединяются к плате управляющего модуля. SOM может выполнять функции хост-машины, а также отчасти выполнять нераспараллеливаемые части программы. К основным

функциям SOM также относится управление внешними устройствами, которые могут быть к нему присоединены как к хост-машине, но главным образом это – управление жёстким диском.

Таким образом, каждый ТМ содержит один SOM, 100 исполнительных модулей (процессоров) и один УП с общей ёмкостью оперативной памяти более 13 гигабайт и 500 гигабайт памяти на жёстком диске. Кроме этого, в состав УМ входит флеш-накопитель ёмкостью 16 Гб, присоединённый к УП и предназначенный для первичной настройки ПЛИС, и два порта USB для связи между ТМ и с внешней средой. Охлаждение ТМ обеспечивается обычными вентиляторами, при включении в систему большого количества ТМ дополнительно может использоваться кондиционер.

Такой ТМ имеет размеры 240 × 220 × 80 мм, что по объёму примерно соответствует двум положенным рядом строительным кирпичам, при этом он имеет производительность – 2 – 6 терафлопс на соответственно 64/32-разрядной арифметике с плавающей точкой (и он уже является вполне самостоятельным, полнофункциональным и законченным многопроцессорным компьютером). Один или несколько ТМ могут составить настольный вариант СДА с производительностью от 2 терафлопс до нескольких десятков терафлопс.

Любое количество ТМ может быть объединено в систему сколь угодно большой производительности, вплоть до сотен петафлопс (изменяются только конструктивные решения и система охлаждения – в этом случае требуемое множество ТМ располагается в соответствующих стойках, а к системе охлаждения обычными вентиляторами добавляется кондиционер), при этом не нужно вносить никаких изменений в программы, операционную систему и другие составляющие системы. То есть любая конфигурация СДА строится из ТМ как любое строение из кирпичиков. Таким образом, вопрос реализации суперкомпьютера с динамической архитектурой любой произвольной мощности является лишь вопросом финансовых вложений. Но в любом случае, при равной производительности по сравнению с наиболее мощными суперкомпьютерами из списка Top-500, СДА будет как минимум на порядок дешевле.

В отличие от ранее реализованных вариантов МДА, в СДА имеется только один управляющий процессор на 100 ИМ, в то время как в прежних вариантах УП имелся в каждом вычислительном модуле. Это связано с тем, что функции УП достаточно просты и при современных характеристиках производительности ПЛИС его мощности вполне достаточно для выполнения своих функций по отношению к указанному множеству ИМ.

Ещё одно отличие СДА от ранее реализованных вариантов МДА заключается в возможности использования для программирования типовых языков программирования, например «С» или FORTRAN, при этом для критических для распараллеливания участков программы создаётся специальная библиотека процедур, которые и выполняются на СДА. Именно таким образом может быть реализована вполне стандартная программа теста Linpack, в котором все некритичные к уровню производительности фрагменты программы выполняются либо на хост-машине, т.е. на обычной ПЭВМ, к которой могут быть подключены и ТМ, либо на SOM, входящих в состав ТМ. Те же фрагменты программы, которые требуют распараллеливания и высокой производительности, будут оформлены в виде соответствующих библиотечных функций, выполняемых на модулях СДА. Они могут быть написаны как на языке ЯРД (и в соответствующем виде загружаться в модули СДА), так и любым другим способом, позволяющим описать программу (процедуру), реализующую соответствующие параллельные вычисления посредством перепрограммирования ПЛИС и выполнения на них соответствующих фрагментов программы.

Заключение

Рассмотренные в данной статье концепция и архитектура СДА позволяют создавать суперкомпьютеры любой вычислительной мощности от 2 – 6 терафлопс до десятков и сотен петафлопс, существенно превышающие по своим параметрам (геометрическому объёму, энергопотреблению, стоимости) существующие суперкомпьютеры при сопоставимой производительности.

При этом обеспечивается высокая надёжность работы системы в целом, независимо от возможных отказов оборудования, независимость программного обеспечения от конкретной конфигурации системы, защищённость системы от несанкционированного доступа, автоматическое распараллеливание программ и другие существенные преимущества.

Литература

1. Торгашев В.А. Средства организации параллельных вычислений и программирования в мультипроцессорах с динамической архитектурой / В.А. Торгашев, И.В. Царев // Программирование. – 2001. – № 4. – С. 53-68.
2. Торгашев В.А. Динамические автоматные сети как модель параллельных вычислений / В.А. Торгашев, И.В. Царев // Вестник компьютерных и информационных технологий. – Москва : Машиностроение. – 2009. – № 3. – С. 11-20.
3. Царев И.В. ЯРД – язык сетевого программирования в распределенных вычислительных системах с динамической архитектурой / И.В. Царев // Искусственный интеллект. – 2008. – № 3. – С. 761-770.
4. Царёв И.В. Программные и апаратные средства коммуникации в мультипроцессорах с динамической архитектурой / И.В. Царёв // Тезисы докладов Международной научной конференции [«Интеллектуальные и многопроцессорные системы – 2004»], (Таганрог ; Донецк). – 2004. – С. 89-92.
5. Царев И.В. Аппаратно-программные методы обеспечения надежности вычислений в мультипроцессорах с динамической архитектурой / И.В. Царев // Известия ТРТУ. Специальный выпуск. – Таганрог : Изд-во ТРТУ, 2005. – С. 61-70.
6. Царев И.В. Информационная безопасность в распределенных вычислительных системах на основе коммуникационных модулей с динамической архитектурой (КМДА) / И.В. Царев // Искусственный интеллект. – 2007. – № 3. – С. 122-130.

В.А. Торгашов, І.В. Царьов

Сім'я суперкомп'ютерів з динамічною архітектурою – концептуальні основи

У статті розглядається архітектура й основні характеристики нового типу суперкомп'ютерів, заснованих на теорії Динамічних Автоматних Мереж (ДАМ) й ідеології раніше розроблених Мультипроцесорів з Динамічною Архітектурою (МДА), що дозволяють конструювати високоефективні та надійні обчислювальні системи у широкому діапазоні продуктивності. Наводяться характеристики деяких технічних і програмних рішень.

V.A. Torgashev, I.V. Tsaryov

Conceptual Basis for a Family of Supercomputers with Dynamic Architecture

Architecture and principal characteristics of the new type of supercomputers based on Dynamic Automata Networks (DAN) theory and on ideology of earlier developed Multiprocessors with Dynamic Architecture (MDA) that permit to construct high-efficient and reliable computational systems in a wide range of performance are considered in the article. Some characteristics of hardware and software decisions are given.

Статья поступила в редакцию 01.06.2009.